TOSCA data types of ASD

tosca.datatypes.asd.extCpdData

```
tosca.datatypes.asd.extCpdData:
    version: 0.1
    derived_from: tosca.datatypes.Root
    description: "Describes the datatype for external connection point definition data"
    properties:
      id:
        description: "The identifier of this extCpdData"
        required: true
        type: string
      description:
        description: >
            This property describes for a particular ExtCpd instance
            what service it exposes.
        required: true
        type: string
      virtual_link_requirement:
        description: >
          Refers in an abstract way to the network or multiple networks that
          the ExtCpd shall be exposed on (ex: OAM, EndUser, backhaul, LI, etc)
        required: true
        type: string
      network_interface_realization_requirements:
        description: >
          Details container implementation specific requirements on
          the NetworkAttachmentDefinition
        required: false
        type: tosca.datatypes.asd.networkInterfaceRequirements
      input_param_mappings:
        description: >
          Information on what helm chart input parameters that
          are required to be configured for this extCpd
        required: false
        type: tosca.datatypes.asd.paramMappings
      resource_mapping:
        description: >
          Kubernetes API resource name for the resource manifest for the service,
          ingress controller or pod
        required: false
        type: string
```

tosca.datatypes.asd.networkInterfaceRequirements

```
tosca.datatypes.asd.networkInterfaceRequirements:
    derived_from: tosca.datatypes.Root
    version: 0.1
    description: "Describes the datatype for network interface requirements"
    properties:
      trunk_mode:
        description: >
               Information about whether the CP instantiated from this Cp is
               in Trunk mode (802.1Q or other). When operating in "trunk mode",
               the Cp is capable of carrying traffic for several VLANs.
               Absence of this property implies that trunkMode is not configured
               for the Cp i.e. It is equivalent to boolean value "false".
        required: true
        type: boolean
        default: false
      ipam:
        description: >
              Identifies whether application expects IP address assignment to be
              managed by the cluster infrastructure (CNI IPAM plugin), or
              configured by orchestrator via for example helm input parameter,
              or if IP assignment is handled by the application itself.
```

```
        required: true
        type: string
        constraints:
          - valid_values: ["infraProvided", "orchestrated", "userManaged"]
        default: "infraProvided"
      interface_type:
        description: >
            Indicates what type of network interface the application expects.
            Kernel based virtual netdev based on CNIs such as ovs | bridge |
            macvlan | ipvlan, or PCIe dev directly visible in application
            namespace with kernel or userspace driver or bonded with the Bond
            CNI, or userspace-CNI based network interface
            (requires DPDK-OVS/VPP vSwitch).
          required: true
        type: string
        constraints:
          - valid_values: ["kernel.netdev", "direct.userdriver", "direct.kerneldriver", "direct.bond",
"userspace"]
        default: "kernel.netdev"
      interface_option:
        description: >
            This attribute describes verified realization options for the
            network interface in question. Currently listed options
            (virtio and memif) are applicable for the interfaceType “userspace”.
        required: false
        type: list
        entry_schema:
              type: string
              constraints:
                  - valid_values: [“virtio", "memif“]
     interface_redundancy:
        description: >
             Identifies switch-plane redundancy method the application uses,
             and that node infrastructure is required to comply with.
             "infraProvided", “left” and “right”: The container sees a
             single vNIC that a) the infrastructure bonds over both switchplanes
             or b) that is connected to the network via only left or
             right the switchplane.
             The other cases are for a mated pair of vnics connecting to
             same network, but where one vNIC connects
             via left switch plane and the other via right switch plane,
             and where the application manages the redundancy.
             "activePassiveBond": the application bonds with move of MAC address.
             "activeActiveBond“: bonded left/right links must be part of a multi-chassis LAG
             "activePassiveL3": application will move application IP address between the vNICs.
             "activeActiveL3": the application uses anycast/ECMP.
        required: true
        type: string
        constraints:
          - valid_values: ["infraProvided", "actPassBond", "actActBond", "actPassL3", "actActL3", "Left",
"Right"]
        default: "infraProvided"
      nic_options:
        description: >
             Identifies for the direct.userdriver interface type, the physical
             nics the driver is verified to work with.
             Allowed values for nic types must be handled via a registry or be standardized.
        required: false
        type: list
        entry_schema:
              type: string
```

tosca.datatypes.asd.paramMappings

```
tosca.datatypes.asd.paramMappings:
    version: 0.1
    derived_from: tosca.datatypes.Root
    description: "Describes the datatype for parameter mapping"
    properties:
```

```
    loadbalancer_IP:
      description: >
            When present, this attribute specifies the name of the deployment
            artifact input parameter through which the orchestrator can
            configure the loadbalancerIP parameter of the K8s service
            or ingress controller that the extCpdData represents.
            Note: The format of the Content strings is specific for each different
            orchestration templating technology used (Helm, Teraform, etc.).
            Currently only a format for use with Helm charts is suggested:
            "<helmchartname>:[<subchartname>.]^(0..N)[<parentparamname>.]^(0..N)<paramname>".
            Whether the optional parts of the format are present depends on how the
            parameter is declared in the helm chart. An example is:
            "chartName:subChart1.subChart2.subChart3.Parent1.Parent2.Parent3.LBIP".
      required: false
      type: string
    external_IPs:
      description: >
            When present, this attribute specifies the name of the deployment
            artifact input parameter through which the orchestrator can
            configure the extermalIPs parameter of the K8s service or ingress
            controller, or the pod network interface annotation, that the
            extCpdData represents.
            Note: The format of the Content strings is specific for each different
            orchestration templating technology used (Helm, Teraform, etc.).
            Currently only a format for use with Helm charts is suggested:
            "<helmchartname>:[<subchartname>.]^(0..N)[<parentparamname>.]^(0..N)<paramname>".
            Whether the optional parts of the format are present depends on how the
            parameter is declared in the helm chart. An example is:
            "chartName:subChart1.subChart2.subChart3.Parent1.Parent2.Parent3.extIP".
      required: false
      type: list
      entry_schema:
         type: string
    nad_names:
      description: >
            Specifies, for an extCpdData respesenting a secondary network interface,
            the name(s) of the deployment artifact input parameter(s) through which
            the orchestrator can provide the names of the network attachment
            definitions (NADs) the orchestrator has created as base for the network
            interface the extCpdData represents.
            Note 1: When the extCpdData represent a networkRedundant/mated-pair of
            sriov interfaces, there are references to 2 or 3 related NADs needed
            to be passed, while for other interface types only one NAD reference
            is needed to be passed.
            Note 2: The format of the Content strings is specific for each different
            orchestration templating technology used (Helm, Teraform, etc.).
            Currently only a format for use with Helm charts is suggested:
            "<helmchartname>:[<subchartname>.]^(0..N)[<parentparamname>.]^(0..N)<paramname>".
            Whether the optional parts of the format are present depends on how the
            parameter is declared in the helm chart. An example is:
            chartName:"subChart1.subChart2.subChart3.Parent1.Parent2.Parent3.nadName".
            Note 3: A direct attached (passthrough) network interface, such as an sriov
            interface, attaches to a network via only one of the two switch planes
            in the infrastructure.
            When using a direct attached network interface one therefore commonly in a
            pod uses a mated pair of sriov network attachments, where each interface
            attaches same network but via different switchplane.
            The application uses the mated pair of network interfaces as a single
            logical “swith-path-redundant” network interface – and this is represented
            by a single extCpdData.
            Also there is a case where a third “bond” attachment interface is used in
            the pod, bonding the two direct interfaces so that the application do not
            need to handle the redundancy issues – application just uses the bond interface.
            In this case, all three attachments are together making up a logical
            “switch-path-redundant” network interface represented by a single extCpdData.
            When three NADs are used in the extCpdData the NAD implementing the bond attachment
            interface is provided through the parameter indicated in the third place in
            the nadNames attribute.
      required: false
      type: list
      entry_schema:
```

```
        type: string
    nad_namespace:
      description: >
            Specifies, for an extCpdData respesenting a secondary network interface,
            the name of the deployment artifact input parameter through which the orchestrator
            can provide the namespace where the NetworkAttachmentDefinitions (NADs) are located.
            Attribute may be omitted if the namespace is same as the application
            namespace.
            Note: The format of the Content strings is specific for each different
            orchestration templating technology used (Helm, Teraform, etc.).
            Currently only a format for use with Helm charts is suggested:
            "<helmchartname>:[<subchartname>.]^(0..N)[<parentparamname>.]^(0..N)<paramname>".
            Whether the optional parts of the format are present depends on how the
            parameter is declared in the helm chart. An example is:
            "chartName:subChart1.subChart2.subChart3.Parent1.Parent2.Parent3.NameSpace".
      required: false
      type: string
```

tosca.datatypes.asd.enhancedClusterCapabilities

```
tosca.datatypes.asd.enhancedClusterCapabilities:
    version: 0.1
    derived_from: tosca.datatypes.Root
    description: "Describes the datatype for parameter mapping"
    properties:
      min_kernel_version:
        description: >
               Describes the minimal required Kernel version, e.g. 4.15.0.
               Coded as displayed by linux command uname –r
        required: true
        type: string
      required_kernel_modules:
        description: >
             Required kernel modules are coded as listed by linux lsmod command,
             e.g. ip6_tables, cryptd, nf_nat etc.
        required: false
        type: list
        entry_schema:
              type: string
      conflicting_kernel_modules:
        description: >
             Kernel modules, which must not be present in the target environment.
             The kernel modules are coded as listed by linux lsmod command,
             e.g., ip6_tables, cryptd, nf_nat etc.
             Example: Linux kernel SCTP module, which would conflict with use of
             proprietary user space SCTP stack provided by the application.
        required: false
        type: list
        entry_schema:
              type: string
      required_custom_resources:
        description: >
            List the custom resource kinds required to be supported in the target
            environment. The list shall include those custom resource kinds which
            are not delivered with the application.
        required: false
        type: list
        entry_schema:
              type: tosca.datatypes.asd.customResourceRequirement
      cluster_labels:
        description: >
             This attribute allows to associate arbitrary labels to clusters.
             These can indicate special infrastructure capabilities (e.g., NW acceleration,
             GPU compute, etc.). The intent of these labels is to serve as a set of
             values that can help in application placement decisions.
             clusterLabels follow the Kubernetes label key-value-nomenclature
             (https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/).
             It is recommended that labels follow a standardized meaning e.g. for node
             features (https://kubernetes-sigs.github.io/node-feature-discovery/v0.9/get-started/features.
html#table-of-contents).
             Example:
             ClusterLabels
               - feature.node.kubernetes.io/cpu-cpuid.AESNI: true
        required: false
        type: list
        entry_schema:
              type: string
      required_plugin:
        description: a list of the name of the required K8s plugin
        required: false
        type: list
        entry_schema:
              type: tosca.datatypes.asd.requiredPlugin
```

tosca.datatypes.asd.customResourceRequirement

```
tosca.datatypes.asd.customResourceRequirement:
       version: 0.1
       derived_from: tosca.datatypes.Root
       description: >
           kind: "Redis", apiVersion: "kubedb.com/v1alpha1"
       properties:
           kind:
              description: "the name of the custom resource requirement"
              type: string
              required: true
           api_version:
              description: "the api version of the custom resource requirement"
              type: string
              required: true
```

tosca.datatypes.asd.requiredPlugin

```
tosca.datatypes.asd.requiredPlugin:
       version: 0.1
       derived_from: tosca.datatypes.Root
       description: "the required K8s plugin"
       properties:
           name:
              description: "the name of the required K8s plugin"
              type: string
              required: true
           version:
             description: "the version of the required K8s plugin"
             type: string
             required: true
```