# Conversion from high-level TOSCA DM to low-level format

## High-level:

The Definition of high-level TOSCA DM about HPA could refer to Supported HPA Capability Requirements(DRAFT). The example, which acts as a resource of SDC,  could refer to [demo.git] / tosca / vCPE /

SDC leverages the resource to set up a service, an example output of the service is attached: service-VcpeWithAll-csar.csar

## Low-level:

There are several components, which acts of SDC client, to parse the output CSAR to do interactive:

- SO:  sends out homing requests to OOF by parsing the CSAR,  with HEAT template,  OOF will respond with the most appropriate flavors accordingly
- Policy: after user distributes the service, it will download the related CSAR and dynamically generate the policies based on specified rules.
- VFC:  sends out homing requests to OOF by parsing the CSAR,  with TOSCA template,  OOF will respond with the most appropriate flavors accordingly

The flow of HPA  like below:

- Policy generates policies after user distributes SDC service by parsing CSAR
- TOSCA: VFC sends homing requests to OOF by parsing the SDC CSAR
- HEAT: SO sending homing requests to OOF by parsing the SDC CSAR
- OOF fetch policies from Policy based on the input from VFC or SO
- OOF fetch flavor from AAI based on the received policies
- OOF composite the output to VFC or SO

We need to keep alignment about the how to parse the SDC CSAR, the mapping from CSAR to low-level fields. below is the logic of Policy side used, the format example refer to OOF R3 HPA & Cloud Agnostic policies

| Policy | Const value or from TOSCA CSAR | Value | Comment |
|---|---|---|---|
| service | Const | "hpaPolicy" | |
| policyName | TOSCA CSAR | "OSDF_CASABLANCA."+ content.getIdentity() | |
| description | Const | "OOF Policy" | |
| templateVersion | | "OpenSource.version.1" | |
| version | | "1.0" | |
| priority | | "5" | |
| riskType | | "Test" | |
| riskLevel | | "2" | |
| guard | | "False" | |
| content.resources | TOSCA CSAR | content.getResources().add(metaData.getValue("name")); | the resource name defined in SDC |
| content.identity | TOSCA CSAR | content.getPolicyType() + "_" + metaData.getValue("name") | |
| content.policyScope | Const<br>TOSCA CSAR | "HPA"<br>sdcCsarHelper.getServiceMetadata().getValue("name"); | List |
| content.policyType | Const | "hpa" | |
| content.flavorFeatures.id | TOSCA CSAR | node.toString | the name of the VDU node |
| content.flavorFeatures.type | Const | "tosca.nodes.nfv.Vdu.Compute" | placeholder |
| content.flavorFeatures.directives.type | | "flavor_directives" | |

| | | | |
|---|---|---|---|
| content.flavorFeatures.directives.attributes.attribute_name | | "flavor_name" | |
| content.flavorFeatures.directives.attribute_value | | "" | |
| content.flavorFeatures.flavorProperties.hpaFeature | TOSCA CSAR | one of available or meaningful values:<br>cpuTopology, basicCapabilities, ovsDpdk, cpuPinning, numa, sriovNicNetwork, pciePassthrough, localStorage, instructionSetExtensions, hugePages based on below Feature judgement flow | based on HPA Policies and Mappings |
| content.flavorFeatures.flavorProperties.mandatory | | from mandatory field | |
| content.flavorFeatures.flavorProperties.architecture | | from hardwarePlatform field | |
| content.flavorFeatures.flavorProperties.hpaVersion | | "v1" | |
| content.flavorFeatures.flavorProperties.hpaFeatureAttributes | | parse and tiny change from configurationValue | |
| content.flavorFeatures.flavorProperties.directives | | [ ] | OOF fill them in |

each VDU maps to a FlavorFeature,

each Resource maps to a Policy which includes a flavorFeatures field,  such field consists of a list of FlavorFeature.

Feature judgement flow, it needs go through all VDUs, CPs and VLs based on the received csar:

First stores all VDUs, CPs, VLs, then:

```
for each (VDU){
    new a FlavorFeature
    if has value under ("virtual_memory#virtual_mem_size or virtual_cpu#num_virtual_cpu"){
        hpaFeature="basicCapabilities "
        generate hapFeatureAttribute based on  value under ("virtual_memory#virtual_mem_size")  and add it into hpaFeatureAttributes
        generate hapFeatureAttribute based on  value under  ("virtual_cpu#num_virtual_cpu") and add it into hpaFeatureAttributes
    }
    if has value under ("virtual_memory#vdu_memory_requirements#memoryPageSize"){
        hpaFeature="hugePages"
        generate hapFeatureAttribute based on  value under  ("virtual_memory#vdu_memory_requirements#memoryPageSize") and add it into hpaFeatureAttributes
    }
    add flavorFeature into the list of FlavorFeatures field.
}
```

```
for each (CP) {
    interfaceType = value under ("virtual_network_interface_requirements#network_interface_requirements#interfaceType")
    if interfaceType == SR-IOV
        hpaFeature="sriovNICNetwork"
    else if interfaceType == PCI-Passthrough
        hpaFeature="pciePassthrough"
    get the generated flavorFeature based on the value under ("virtual_binding")

    get the FlaovrProperties from flavorFeature

    new a FlavorProperty

    generate hapFeatureAttribute based on  value under ("virtual_network_interface_requirements#nic_io_requirements#logical_node_requirements#pciVendorId") and add it into hpaFeatureAttributes

    generate hapFeatureAttribute based on  value under ("virtual_network_interface_requirements#nic_io_requirements#logical_node_requirements#pciDeviceId") and add it into hpaFeatureAttributes

    generate hapFeatureAttribute based on  value under ("virtual_network_interface_requirements#nic_io_requirements#logical_node_requirements#pciNumDevices") and add it into hpaFeatureAttributes

    if has value under ("virtual_network_interface_requirements#nic_io_requirements#logical_node_requirements#physicalNetwork")

        get the info of node, which is a Virtual Link, based on the value under ("virtual_link") , it is stores in VLs

        get the property value under ("physicalNetwork")

        generate hapFeatureAttribute based on  value under ("virtual_network_interface_requirements#nic_io_requirements#logical_node_requirements#physicalNetwork") and add it into hpaFeatureAttributes

    add FlavorProperty into flavorFeature.getFlavorProperties()

}
```