ONAP managed Object & VNF CLI

Kanagaraj.Manickam@huawei.com
Managed Object - definition

Managed object

From Wikipedia, the free encyclopedia

In telecommunication, the term managed object has the following meanings:

1. In a network, an abstract representation of network resources that are managed. With "representation", we mean not only the actual device that is managed, but also the device driver, that communicates with the device. An example of a printer as a managed object is the window that shows information about the printer, such as the location, printer status, printing progress, paper choice, and printing margins.

The database, where all managed objects are stored, is called Management Information Base. In contrast with a CI, a managed object is "dynamic" and communicates with other network resources that are managed.

Note: A managed object may represent a physical entity, a network service, or an abstraction of a resource that exists independently of its use in management.

2. In telecommunications management, a resource within the telecommunications environment that may be managed through the use of operation, administration, maintenance, and provisioning (OAMP) application protocols.

© This article incorporates public domain material from the General Services Administration document "Federal Standard 1037C"© (in support of MIL-STD-188).
What is ONAP managed object (OmO)

In ONAP architecture, User interface is provided by ONAP portal, UUI and ONAP CLI projects, which communicate with different internal ONAP services such as APPC, SDNC, Policy, MultiVM, VFC, AAI, SDC, SO, etc, over the REST API with its own internal models defined by these services. Those models are specific to the services having both domain model attributes and implementation specific attributes, which are not in the interest of user. And there are two issues:

1. All 3 user interfacing projects directly integrate with respective services and provide different view of internal model provided by ONAP services. This bring inconsistent user experience
2. Any change in internal API is directly exposed to these user interfacing project, end up modifying them, even the change is not related to user interface attribute of model.

ONAP CLI models user-facing view of ONAP models from each of the ONAP services' internal models and this view captures only those attributes, which are domain and in the interest of user. This user-facing model view is called ONAP managed object (OmO).

- OmO = \{ONAP service Internal model attributes\} minus \{Implementation specific attributes\} = Domain model
- So All attributes in OmO are user-facing and domain-model specific.
Available ONAP managed object (OmO)

ONAP CLI provides following managed objects:

| cloud   | aai | STABLE | a cloud region in Onap |
| complex | aai | STABLE | a cloud complex in Onap |
| csar    | vnf-sdk | STABLE | Uploads the CSARs in marketplace |
| customer | aai | STABLE | a customer in Onap |
| ems     | aai | STABLE | a EMS in Onap |
| microservice | msb | STABLE | microservice into Onap |
| policy  | policy | STABLE | a policy in PAP |
| policy-type | policy | STABLE | a policy type |
| sdcn    | aai | STABLE | a SDNC in Onap |
| service | so  | STABLE | a service instance using SO |
| service-model | sdc | STABLE | Service model in SDC |
| service-type | aai | STABLE | Add a service type in Onap |
| subscription | aai | STABLE | a subscription of a customer for given |
| tenant  | aai | STABLE | a tenant under given cloud region in Onap |
| vnf-model | sdc | STABLE | Virtual function from Vendor Software |
| vnf-module | so  | STABLE | a VNF Module |
| ns-catalog | vfc | STABLE | vfc ns |
| vnf-catalog | vfc | STABLE | vfc vnf |
| vfc-smc | vfc | STABLE | vfc-smc ns |
| vim     | aai | STABLE | a VIM under a given cloud region |
| vim-agreement | sdc | STABLE | License agreement |
| vim     | sdc | STABLE | License Model |
| vim-entitlement-pool-cm | sdc | STABLE | Entitlement Pool |
| vim-feature-group-crest | sdc | STABLE | feature group Pool |
| vim-key-group | sdc | STABLE | License Key Group |
| vnf     | so  | STABLE | a VNF |
| vnf-m | aai | STABLE | a VNF in Onap |
| wsp     | sdc | STABLE | Vendor Software Product |

---

ONAP Internal services:
How to Leverage OmO in across ONAP accessibility points
Demo | OVP in Action

✓ **OCOMP unifies the model and actions** as ONAP managed object (using OCS YAML)
  - same user experience across all ONAP services, Network services, VNF and use-cases.
✓ **Develop once, share across roles** Certification, testing, devops and production/automation.
  - Saves everyone from duplicate effort spent on developing their own copy of same automation
✓ **Every action available as its to different role**
  - as command from CLI console (devops)
  - as test case from VTP (testing, certification, production)
✓ **Tester fills the user role** in the absence of user
  - All roles see same behavior for a given action

<table>
<thead>
<tr>
<th>Role</th>
<th>Operator/Vendor</th>
<th>OVP lab</th>
<th>CI/CD</th>
</tr>
</thead>
<tbody>
<tr>
<td>As Command using CLI</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>As Test case using VTP</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

6X vs 1X

* ONAP managed object - ONAP CLI commands
# Open CLI Platform (OCLIP)
ONAP VNF CLI
Requirements

- **REQ1**: As a operator, After ONAP provisioned the Network service and it’s VNFs, I would like to check the health status of every VNF

- **REQ2**: As a vendor, When I validate the VNF life-cycle in ONAP environment, I want to check the VNF health status

- **REQ3**: As a system, Once the Network service is provisioned, Before providing the access details to the end-user, It needs to check the status of the all VNFs provisioned as part of the service activation. (by both SO and VFC)
ONAP supports 2 scenarios

- ONAP supports two service provisioning scenarios:
  - HEAT based: SDC + SO + Multi-VIM + SDNC/APPC
  - TOSCA based: SDC + VFC + Multi-VIM [+ VNFM driver]
- Both scenario uses AAI as the common inventory.
- APPC & VNFM driver act as mgmt interface for VNF provisioned by ONAP
ONAP CLI for health check and solution for requirements

Vnf-health:
Show the VNF health status for the given service-instance id.

<table>
<thead>
<tr>
<th>output</th>
<th>usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vnf-id</td>
<td>VNF ID</td>
</tr>
<tr>
<td>vnf-name</td>
<td>VNF name</td>
</tr>
<tr>
<td>Status</td>
<td>ACTIVE/IN-ACTIVE</td>
</tr>
</tbody>
</table>
Thank you
HEAT scenario Analysis

AAI: /aai/v14/nodes/service-instances/service-instance/service-instance-1
{
  "service-instance-id": "service-instance-1",
  "service-instance-name": "vlb_si_ewm_1",
  "relationship-list": [{
    "relationship": [{
      "related-to": "generic-vnf",
      "relationship-label": "org.onap.relationships.inventory.ComposedOf",
      "related-link": "/aai/v14/network/generic-vnfs/generic-vnf/generic-vnf-1"
    }]
  }
}

AAI: /aai/v14/network/generic-vnfs/generic-vnf/generic-vnf-1
{
  "vnf-id": "generic-vnf-1",
  "vnf-name": "generic-vnf-one",
  "vnf-type": "vFWDT 2019-05-14 21:39:/vFWDT_vPKG f9ca4e08-2039 0",
  "service-id": "4ac68777-580d-4c77-be9e-715af194eff2",
  "prov-status": "ACTIVE",
  "orchestration-status": "Active",
  "relationship-list": [{
    "relationship": [{
      "related-to": "vserver",
      "relationship-label": "tosca.relationships.HostedOn",
      "related-link": "/aai/v14/cloud-infrastructure/cloud-regions/cloud-region/CloudOwner/RegionOne/tenants/tenant/09d8566ea4e43aa974cf447ed591d77/vservers/vserver/47df73c0-e35d-488d-864e-53bf103a0c83"
    }]
  }
}

AAI: /aai/v14/cloud-infrastructure/cloud-regions/cloud-region/CloudOwner/RegionOne/tenants/tenant/09d8566ea45e43aa974cf447ed591d77/vservers/vserver/47df73c0-e35d-488d-864e-53bf103a0c83
{
  "l-interfaces": {
    "l-interface": [{
      "l3-interface-ipv4-address-list": [
        "10.0.136.7"
      ]
    }
  }
}

APPC: /restconf/operations/appc-provider-lcm:health-check
{
  "input": {
    "common-header": {
      "timestamp": "{{serverDate}}",
      "api-ver": "2.00",
      "originator-id": "onap-cli",
      "request-id": "uuid-1",
      "action": "HealthCheck",
      "action-identifiers": {
        "vnf-id": "generic-vnf-1"
      }
    },
    "payload": {
      "request-parameters": {
        "vnf-name": "generic-vnf-one",
        "host-ip-address": "10.0.136.7",
        "configuration-parameters": {
          "vnf-name": "generic-vnf-one"
        }
      }
    }
  }
}